Trabajo Fin de Grado
Ingeniería Electrónica, Robótica y Mecatrónica

Construcción de un huerto automatizado

Autor: Jaime García Ramírez

Tutor: Alfredo Pérez Vega-Leal

Dpto. Ingeniería Electrónica
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2024

Trabajo Fin de Grado
Ingeniería Electrónica, Robótica y Mecatrónica

Construcción de un huerto automatizado

Autor:
Jaime García Ramírez

Tutor:
Alfredo Pérez Vega-Leal
Profesor Titular

Dpto. Ingeniería Electrónica
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2024

Trabajo Fin de Grado: Construcción de un huerto automatizado

Autor: Jaime García Ramírez
Tutor: Alfredo Pérez Vega-Leal

El tribunal nombrado para juzgar el trabajo arriba indicado, compuesto por los siguientes profesores:

Presidente:

Vocal/es:

Secretario:

acuerdan otorgarle la calificación de:

El Secretario del Tribunal

Fecha:

Agradecimientos

La culminación de este trabajo supone el fin de una etapa cuyo inicio se me hace muy lejano, pero a la vez muy reciente. Parece que fue ayer cuando llegué a la ETSI, cargado de miedos pero también de esperanzas, de temor a no ser capaz de superar los desafíos que se encontraban frente a mi, a fracasar, a decepcionar a todos. Pero hoy estoy aquí, este trabajo es la prueba de que pude lograrlo, de que superé el camino, pero también es la prueba de que estoy rodeado de muchas personas maravillosas que me han apoyado en cada uno de mis pasos, por eso no sólo este trabajo, si no todo este recorrido va dedicado a ellos, a mis padres, que me han apoyado desde el minuto uno y me han dado fuerzas para continuar cuando yo mismo no me veía capaz de lograrlo, a mi pareja, que se unió en el trayecto y me ha dado los momentos de paz que necesitaba y todo su apoyo, a mis amigos, por tantas tardes de estudio, prácticas y ratitos que hicieron mas ameno el peso de los exámenes, a mis profesores que me han guiado a cada paso, en especial a Alfredo, mi tutor en este proyecto, a mi tío Antonio, que me ha dado la posibilidad de construir mi sistema, a mis abuelos, tanto los que están aquí como los que ya nos han dejado, que me hacen saber lo orgullosos que están de mí cada dia, y por supuesto a toda mi familia, que siempre han estado ahí para mi, por todo esto solo puedo deciros una cosa.

Gracias.

Jaime García Ramírez
Sevilla, 2024

Resumen

El objetivo de este trabajo es el diseño y construcción de un modelo a escala de un huerto circular que realice diversas funciones de manera autónoma, como bien pueden ser el monitoreo de la temperatura del terreno o el riego de este mismo.

Para satisfacer las necesidades del sistema, se utilizará un microprocesador de bajo coste con interfaz *Wi-Fi* que gestione el funcionamiento del resto de elementos que lo conforman y subirá la información a la red para ser consultada de forma remota.

Por último, se considerarán las posibles opciones de mejora que tendría el diseño de cara a crear otras versiones futuras que contemplen funcionalidades añadidas.

Abstract

The objective of this project is the design and construction of a scale model of a circular garden that perform various functions autonomously, such as temperature monitoring of the terrain or its irrigation.

To meet the needs of the system, a low-cost microprocessor with *Wi-Fi* interface that manages the operation of the rest of the elements that make it up and will upload the information to the network to be consulted remotely.

Finally, the possible improvement options that the design would have will be considered in order to create other future versions that include added functionalities.

Índice

1 Introducción

El placer más noble es el júbilo de comprender

Leonardo Da Vinci

A lo largo de la historia, el ser humano ha desarrollado numerosos avances tecnológicos con el objetivo de aumentar la producción agrícola.

Frente al panorama que afrontamos actualmente, y de cara al futuro cercano, es evidente que será necesario una vez más poner a prueba nuestro ingenio con el objetivo de suplir las necesidades de la población mundial en cuestión de alimentos que se aproximan en los próximos años, lo que planterá numerosos retos en diversos campos de investigación.

1.1 Contexto y Objetivos

Se estima que para el año 2050 la población mundial alcanzará aproximadamente los 9.700 millones de habitantes, lo cual supone un incremento alrededor del 25 % con respecto a la población actual. Este aumento se verá reflejado principalmente en países en vía de desarrollo como México, China o India, entre otros, el cual sumado al aumento de la calidad de vida y del porcentaje poblacional que se espera habite en areas urbanas, que se estima alcance un 70 % de la población total del planeta para el año 2050, supondrá duplicar la producción de alimentos actual. [1]

Este proyecto se enmarca dentro de tal ámbito, buscando desarrollar un sistema que permita monitorear de cerca las condiciones a las cuales son sometidas las plantas siendo cultivadas, y así obtener una producción óptima de los diferentes alimentos que estas puedan ofrecer, mientras reduce el esfuerzo que el agricultor deba realizar, e intentando amenizar su tarea dentro de lo posible.

Además, también buscamos que este sistema sea lo más compacto y económico posible, con el objetivo de suplir las futuras necesidades que los países anteriormente nombrados vayan a tener, y haciéndolo adecuado para ser instalado en los pequeños espacios que puedan estar disponibles en una ciudad, como puedan ser terrazas o patios interiores. Aun así, cabe recalcar que el producto mencionado tendría unas dimensiones mayores que el explicado en estas páginas, cuyo único propósito es reflejar el funcionamiento y el aspecto que podría tener el producto final, siendo a la vez portable para la realización de demostraciones en vivo.

1.2 Diseño de Concepto

Para que el modelo del huerto automatizado cumpla los requisitos anteriormente expuestos, se ha propuesto como diseño un terreno circular en cuyo centro se situe una estructura similar a una grúa capaz de abarcar el terreno nombrado anteriormente gracias a su capacidad de girar y de alargar o retraer un "brazo". En la punta de esta estructura se situará la salida del riego, de forma que podamos regar toda la superficie con una sola fuente de agua, además de un sensor que medirá la temperatura del terreno a distancia.

Con el fin de hacerlo portátil, el terreno circular descrito anteriormente tendrá un tamaño limitado, y se situará sobre un espacio donde se resguardará un depósito de agua, así como toda la electrónica que controla el huerto y no deba actuar directamente sobre este.

Por último, añadir que este dispositivo busca ser *Plug and Play*, de forma que la única intervención que el usuario deba realizar, a parte de proporcionar los datos de su red *Wi-Fi*, sea conectarlo ,o desconectarlo, a una toma de corriente.

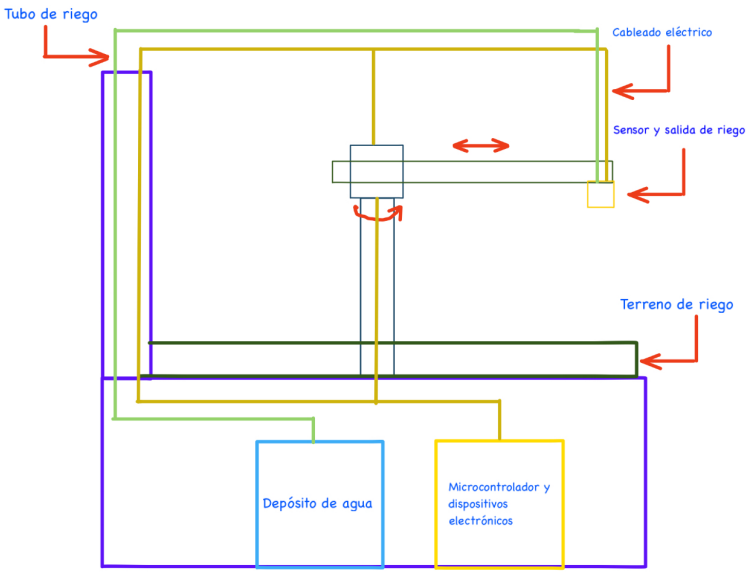

Figura 1.1 Boceto de Huerto Automatizado.

1.3 Hardware

En esta sección se listarán cada uno de los dispositivos electrónicos utilizados para la construcción del sistema y se explicará cómo funcionan.

1.3.1 Feather HUZZAH ESP8266

El microcontrolador a utilizar para el desarrollo de este proyecto es el Feather Huzzah ESP8266, fabricado por la empresa Adafruit.

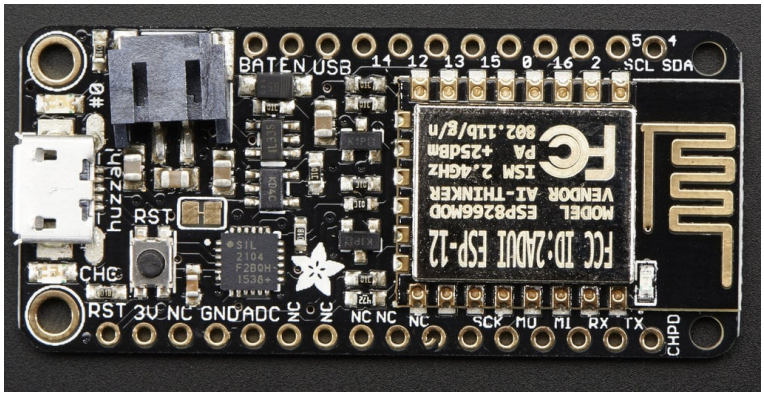

Figura 1.2 Feather HUZZAH ESP8266 [2].

Este microcontrolador cuenta con todo lo necesario para abordar las funciones que debe desempeñar durante el funcionamiento de nuestro sistema, contando además con un precio reducido, lo que lo convierte

en una opción ideal teniendo en cuenta nuestros objetivos.

El ESP8266 incluye por defecto un adaptador *Wi-Fi* y es compatible con los protocolos IP y TCP, lo que nos permitirá convertir nuestro huerto en un dispositivo IoT [3] y subir nuestros datos a la red para ser consultados remotamente.

Además de su predisposición para conetarse a Internet, también cuenta otras características que nos permitirán conectar, controlar y alimentar el resto de dispositivos que conforman el sistema, las más relevantes de estas características son las siguientes:

Tabla 1.1 Características Feather HUZZAH ESP8266 [2].

Especificaciones	
Frecuencia de reloj	80MHz o 160 MHz
Memoria Flash	4 MB
Pines GPIO	9
Protocolos de comunicación	I²C,SPI, UART y*Wi-Fi*
Voltaje de señales lógicas	3.3V
Pines de salida de Voltaje	3.3V, 5V 3.7V[a]

[a] Para disponer de este pin de salida es necesario conectar una batería de litio externa

Una vez conocido de lo que es capaz este microcontrolador, debemos ser conscientes de qué puertos nos dan acceso a sus diferentes funcionalidades, además de tener en cuenta que ciertos pines,como los pines 0, 2 y 15, no pueden ser usados como entradas, ya que tienen conectados a ellos resistencias internas que pueden intervenir con las medidas en el caso de los pines 2 y 15, o tienen otras limitaciones en el caso del pin 0 [2].

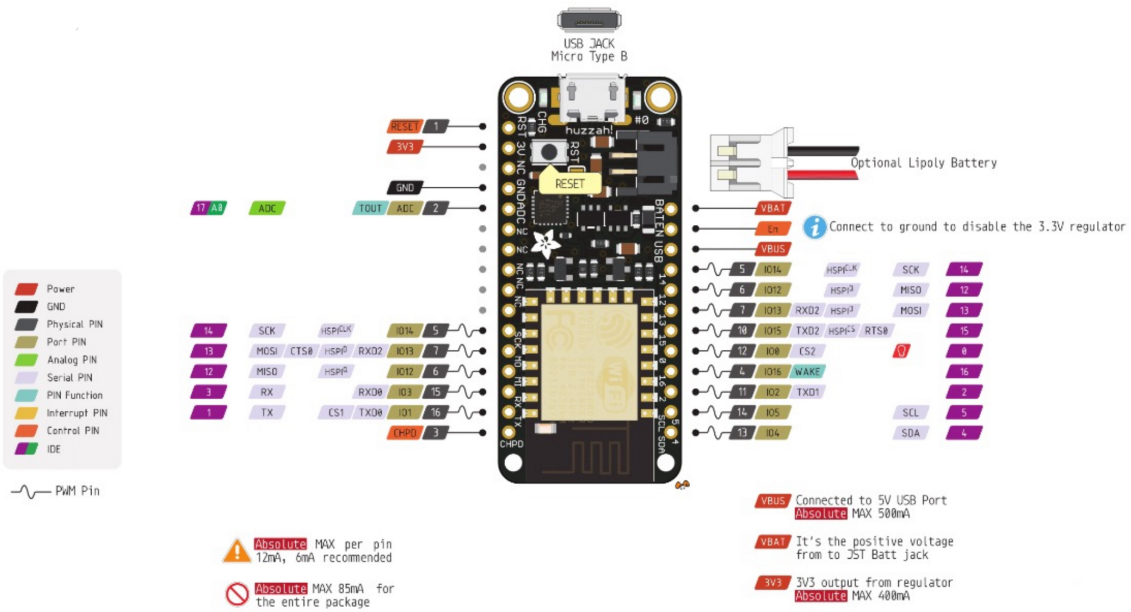

Figura 1.3 Pinout Feather HUZZAH ESP8266 [2].

1.3.2 Servomotores Analógicos de Rotación Continua

Los encargados de dotar de movimiento a todo el mecanismo que acciona el huerto son los servomotores de rotación continua fabricados por Multicomp Pro.

A diferencia de los servomotores convencionales, estos no se enuentran limitados a un rango de 180º grados, lo que nos permite realizar acciones sobre una zona mayor que si utilizaramos su homólogo universal,

Figura 1.4 Servomotor Analógico de Rotación Continua [4].

además estos motores cuentan con la posibilidad de regular la velocidad y el sentido en el que giran, lo que los vuelve una opción casi perfecta para las funciones que deben realizar en nuestro proyecto.

Por otro lado, estos cuentan con una desventaja respecto a los ya mecionados servomotores clásicos, a diferencia de estos, los servomotores de rotación continua funcionan en bucle abierto, lo que significa que no podemos conocer en qué posición se encuentra situado este, ni fijarlo en una posición específica, por lo que será necesario informar de otra forma al microcontrolador de que se ha alcanzado la posición deseada. [5]

Dependiendo de si conectamos el servomotor al pin de alimentación de 5V proporcionado por el microcontrolador, o a una fuente externa que lo alimente a 6V, conseguiremos diferentes valores en el desempeño:

Tabla 1.2 Especificaciones Eléctricas Servomotor Analógico de rotación continua a diferentes voltajes [4].

Especificaciones	4.8V	6V
Corriente de Reposo	5mA	6mA
Velocidad sin carga	110RPM	130RPM
Corriente en funcionamiento(sin carga)	100mA	120mA
Par máximo de Parada	1.3Kg-cm	1.5Kg-cm
Corriente crítica	550mA	650mA

El control sobre el sentido de giro y la velocidad a la que se realiza es análogo a controlar la posición de un servomotor convencional. Este se realiza a través de señales PWM de 50Hz que varían de 1 a 2 ms. Detener el servomotor es igual a colocar uno de los servomotores convencionales en una posición de 90º[1], mientras que valores de PWM que situarían al servomotor convencional en ángulos menores a este, supondrán que el servomotor continuo gire en el sentido de las agujas del reloj, aumentando su velocidad hasta alcanzar el máximo en un valor equivalente a 0º, de igual forma, para girar en sentido antihorario debemos proporcionar pulsos equivalentes a ángulos mayores a 90º, alcanzando la máxima velocidad en este sentido en 180º. [5]

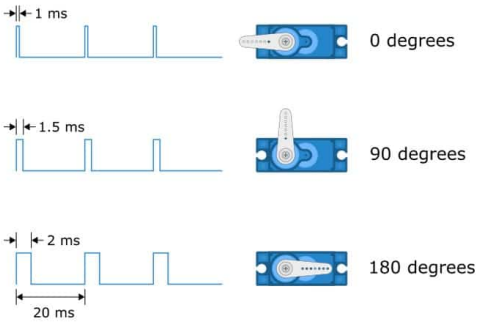

Figura 1.5 Control de un servomotor convencional a través de pulsos PWM [5].

[1] En la práctica, este valor puede encuentrarse desplazado dependiendo del servomotor

1.3.3 Termómetro Infrarrojo MLX90614

Como ya se ha comentado en apartados anteriores, deseamos medir la temperatura a la que se encuentra el terreno sin interactuar con él, función para la que un termómetro infrarrojo compacto y sin un elevado precio como este es perfecto.

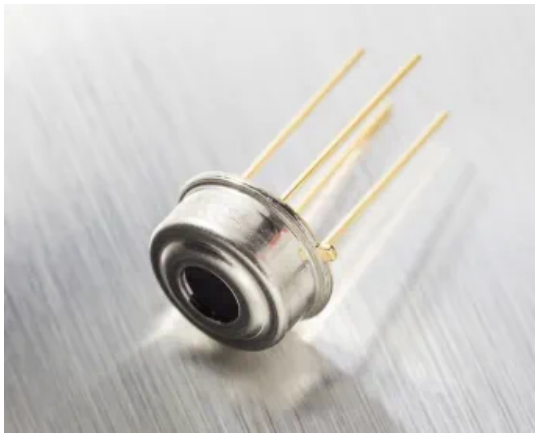

Figura 1.6 Termómetro Infrarrojo MLX90614 [6].

El funcionamiento de estos termómetros se basa en que mientras mayor sea la temperatura de un objeto, mayor será el movimiento de sus partículas, lo que producirá que emitan una mayor cantidad de luz infrarroja, estos dispositivos son capaces de medir esa radiación y convertirla en temperatura. El termómetro infrarrojo calcula la diferencia entre la radiación IR que emite el objeto a medir y la del entorno, a partir de esa diferencia calcula la temperatura del objeto. [7]

Existen diferentes modelos dentro de esta familia de dispositivos, nosotros utilizaremos el modelo BAA, que admite un rango de temperaturas entre -40ºC a 125ºC para la temperataura ambiente, y desde -70ºC hasta 380ºC en el caso de la temperatura de la superficie a medir. [6]

Dependiendo del valor de estas temperaturas, la precisión del valor arrojado por el termómetro puede variar drásticamente.

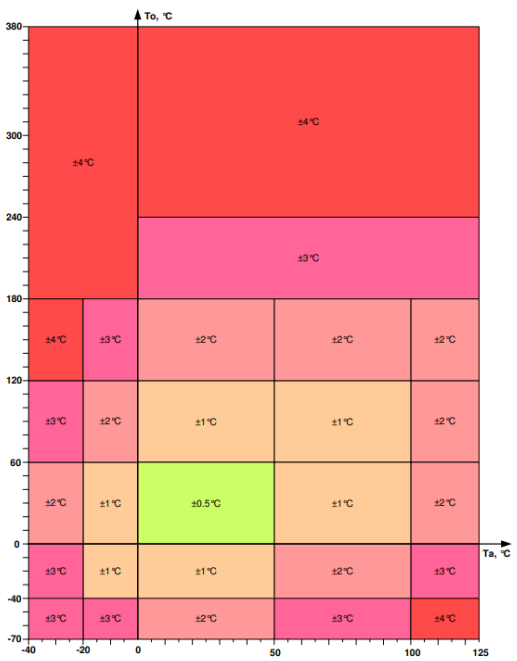

Figura 1.7 Precisión del MLX90614 en función de la Temperatura ambiente y de la Temperatura de la superficie a medir [6].

Afortunadamente, nuestras temperaturas siempre se encontrarán dentro de un rango en el cual el termómetro infrarrojo ofrece la mayor precisión con la que es capaz de medir, por lo que los valores que obtengamos tendrán un error de $\pm$ 0.5ºC. [6]

El sensor puede ser alimentado con valores de entre 2.6V hasta 3.6V, siendo lo convencional 3V [6], sin embargo, el modelo que nosotros tenemos también admite alimentarse a 5V gracias a que posee un regulador de tensión incorporado [7].

Otro punto a tener en cuenta con respecto a la utilización de este sensor es el campo de visión(FOV) y la distancia del objeto a medir. El FOV es un ángulo en el que se promedian todas las temperaturas que detecta el sensor, si el objeto a medir no lo ocupa por completo, el resultado será el promedio de las temperaturas pertenecientes a los objetos que se encuentren dentro del FOV, cuanto más cerca estemos del objeto cuya temperatura deseemos calcular, menos probable será que esto ocurra [7]. Por suerte, en nuestro caso deseamos obtener la temperatura del terreno, por lo que prácticamente siempre el FOV será totalmente ocupado por el objeto que deseamos medir.

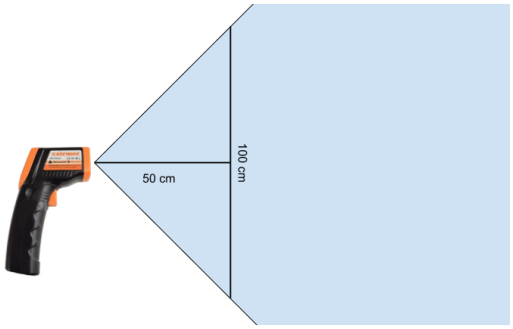

Figura 1.8 FOV de un termómetro infrarrojo [7].

Por último, cabe resaltar que la comunicación entre el sensor y el microcontrolador se realiza a través del protocolo I²C, el cuál será comentado en capítulos posteriores.

1.3.4 RTC DS3231

Para que nuestro sistema sea consciente de cuándo debe regar, será necesario que conozca qué hora es en todo momento, el encargado de suplir esta necesidad será el RTC DS3231.

Figura 1.9 RTC Ds3231 [8].

Este tipo de componentes hacen uso de un oscilador de cristal, el cuál se ajusta por sí solo a los cambios de temperatura , para calcular el paso del tiempo. Dichos dispositivos funcionan a una frecuencia de 32 kHz, la misma que usan los relojes de cuarzo y otros relojes similares. Además, el propio RTC tiene en cuenta el final de los meses con una cantidad de días menor a 31 de forma automática. [9]

La ventaja de usar un RTC frente a los temporizadores internos que los microcontroladores suelen incluir es que estos cuentan con una alimentación externa, usualmente una batería de litio, aunque también se usan supercondensadores en modelos más novedosos, gracias a los cuales son capaces de seguir calculando la hora y fecha actual a pesar de que el sistema se desconecte de la corriente, evitando que pierdan la referencia de tiempo y la hora se resetee. [8]

Figura 1.10 Pila de litio de 3V [10].

El modelo DS3231, el cuál vamos a utilizar, tiene ciertas ventajas en cuanto a precisión frente a otros modelos como el DS1307, sobretodo a temperaturas elevadas. También cuenta con otras funcionalidades, como la capacidad de generar una onda cuadrada de varias frecuencias que puede servir como señal de reloj para otros componentes, una memoria EEPROM y una alarma, la cual otros modelos no poseen, aunque ninguna de estas son de utilidad en este proyecto. Al conjunto de todo el circuito integrado se le conoce como ZS-042. [8]

Al igual que el otro sensor, este también hace uso del protocolo I²C, por lo que deberán compartir el bus de comunicación.

1.3.5 Bomba de agua JT-DC3L

Es evidente que de alguna forma debemos transportar el agua desde nuestro depósito hasta el terreno para que este pueda ser regado, para ello usaremos la bomba de agua JT-DC3L.

Figura 1.11 Bomba de agua JT-DC3L [11].

Este es uno de los elementos más simples del proyecto, simplemente cuenta con un cable positivo y otro negativo, al conectarlos, la bomba de agua empezará a funcionar inmediatamente. Es importante recalcar que no se debe activar cuando esta se encuentra fuera del agua, si esto ocurriera la bomba se fundiría, por lo que es vital evitar que el depósito se encuentre totalmente vacío a la hora de regar. [11]

Las principales características de este dispositivo son las siguientes:

Tabla 1.3 Características Bomba de agua JT-DC3L [11].

Especificaciones	
Voltaje de entrada	2,5-6 V
Corriente	130-220 mA
Flujo	80-120 L/h
Altura máxima de bombeo	1 m

Como este dispositivo no cuenta de ningún tipo de control para activar o detener su funcionamiento, será necesario hacer uso de otro elemento para realizar este trabajo, ya que los pines del microcontrolador no son capaces de suministrarle suficiente corriente y voltaje a la bomba.

1.3.6 Relé de 5V

El dispositivo que realizará la tarea comentada en el apartado anterior será un relé de 5V, lo que nos permitirá alimentar a la bomba directamente desde el pin de 5V suministrado por el microcontrolador.

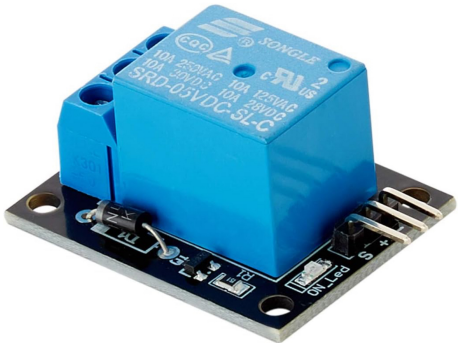

Figura 1.12 Relé de 5V [12].

Un relé es un aparato eléctrico que funciona como un interruptor, abriendo y cerrando el paso de la corriente eléctrica, pero accionado eléctricamente. Existen diferentes tipos de relés, los más comunes y utilizados, y a los cuáles pertenece este, son los relés electromecánicos. Estos están compuestos por un electroimán, que al ser excitado, atrae a la armadura, cerrando o abriendo los contactos dependiendo de si estos son NA o NC. [13]

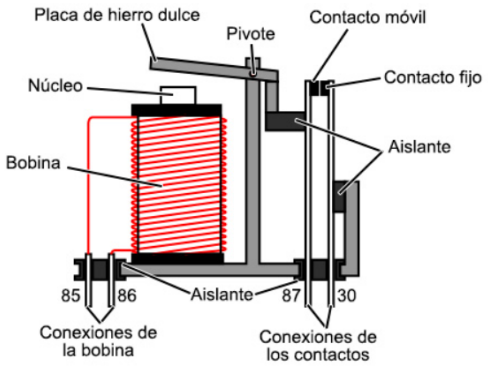

Figura 1.13 Esquema de un relé electromecánico [14].

En nuestro caso, la bomba se conectará al contacto NC, de forma que se active cuando se cierre el relé.

1.3.7 Otros Dispositivos

Para el desarrollo de este proyecto también se ha hecho uso de otros elementos, los cuales se listan a continuación:

- 1 Protoboard para alojar el resto de componentes.

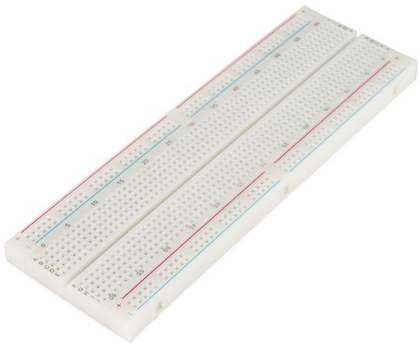

Figura 1.14 Protoboard [15].

- 2 resistencias de 10 KΩ.

Figura 1.15 Resistencias de 10 KΩ. [16].

- 2 finales de carrera.

Figura 1.16 Final de Carrera [17].

- Cables Macho-hembra, Macho-macho y Hembra-hembra.

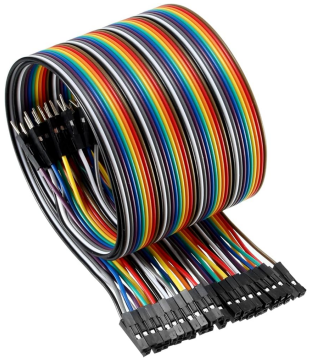

Figura 1.17 Cables Macho-hembra [18].

- 1 Portapilas de 4 pilas de 1.5 V.

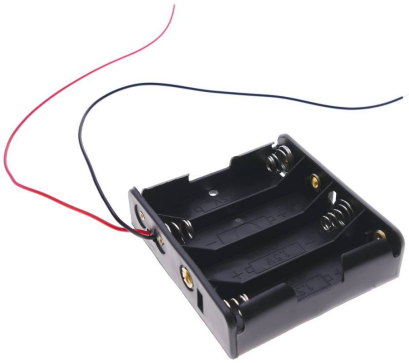

Figura 1.18 Portapilas de 4 pilas [19].

1.4 Software

En esta sección se comentarán las diferentes plataformas que se usaron de cara a programar el sistema y representar sus resultados.

1.4.1 Arduino IDE

Arduino IDE es un editor de texto para escribir código desarrollado por la empresa del mismo nombre, además de esto, también cuenta con otras herramientas que pueden ser útiles a la hora de programar código, como son una consola de texto donde se reportarán los errores o *warnings* que tenga el programa, o un monitor serie para observar lo que nuestro microcontrolador retransmite a través de su puerto serie, el cuál será nuestra principal forma de comunicación con él. [20] Este software ha sido específicamente desarrollado para las placas propias de Arduino, sin embargo, haciendo uso del paquete de placas ESP8266 es posible programar nuestro microcontrolador con este programa. [2]

La sencillez de su lenguaje junto a la gran comunidad que lo respalda, convierte a Arduino IDE en un entorno ideal para el desarrollo de proyectos como este, cuya programación se simplifica gracias a las numerosas librerías que podemos encontrar, de las cuáles nosotros usaremos las siguientes:

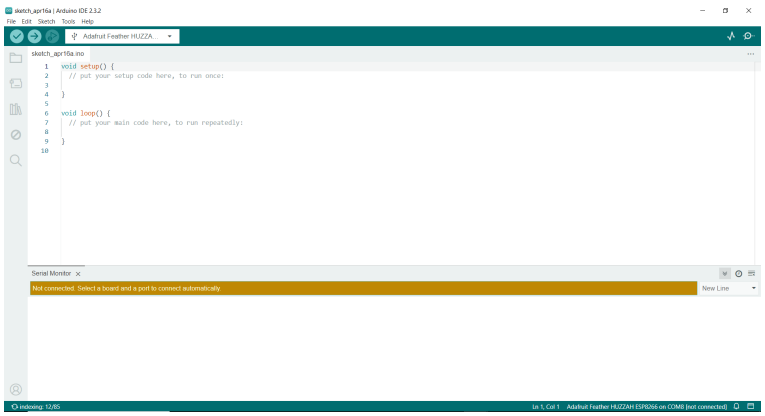

Figura 1.19 Interfaz de Arduino IDE.

- **Wire.h**: Permite al microcontrolador comunicarse con dispositivos que usan el bus I²C.
- **RTClib.h**: Proporciona comandos para el uso del módulo RTC.
- **ThingSpeak.h**: Añade instrucciones para la publicación de datos en ThingSpeak.
- **Adafruit_MLX90614.h**: Cuenta con comandos que facilitan el uso del termómetro infrarrojo MLX90614.
- **ESP8266WiFi.h**: Esta librería posibilita que el microcontrolador se conecte a una Red *Wi-Fi*.
- **Servo.h**: Gracias ella podemos controlar los servomotores de forma sencilla.

1.4.2 ThingSpeak

ThingSpeak es una plataforma de análisis de dispositivos IoT desarrollada por MathWorks, creadores de Matlab. ThingSpeak permite subir a la nube datos en vivo para ser visualizados o analizados de forma sencilla sin necesidad de construir servidores o desarrollar software web. Esta plataforma almacena los datos en canales, que permiten crear hasta 8 campos de datos por canal, en nuestro caso sólo se usará uno, pero esto nos da la posibilidad de añadir futuros sensores que obtengan otro tipo de datos sin necesidad de migrar la forma en la que son almacenados. [21]

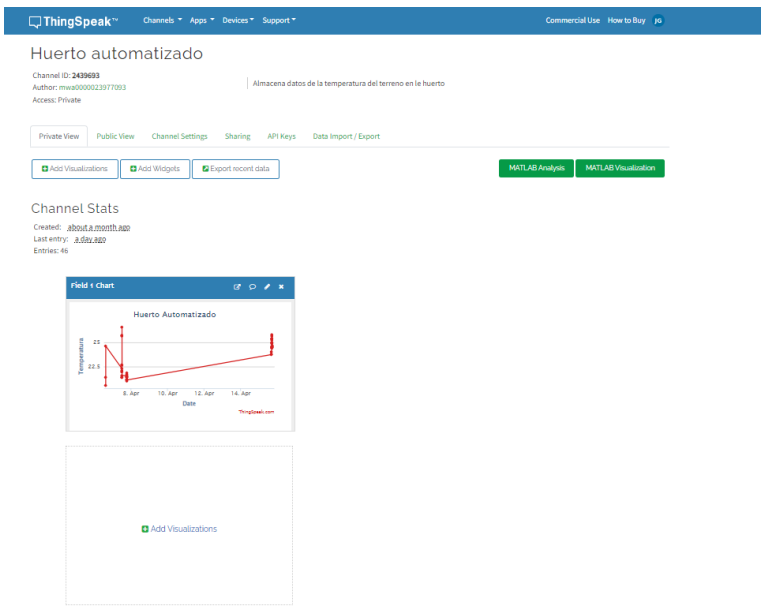

Figura 1.20 Canal de ThingSpeak perteneciente al proyecto.

Thingspeak permite acceder a los canales desde cualquier navegador web, lo cual posibilita que los datos puedan ser fácilmente consultados desde cualquier dispositivo con una conexión a Internet, supliendo de esta forma nuestra necesidad de consultar la información desde cualquier punto.

2 Comunicaciones

Aquel que lo intentó y no lo consiguió es superior al que ni lo intentó.

Arquímedes de Siracusa

En este capítulo se listarán los diversos protocolos de comunicación utilizados durante el desarrollo del proyecto para interconectar y transferir datos entre el microcontrolador y los múltiples sensores utilizados o ThingSpeak, además de ilustrar su funcionamiento.

2.1 I²C

I²C hace referencia a un bus de comunicaciones serie, diseñado por Philips, habiendo lanzado diferentes versiones desde el año 1982. Este cuenta con varias tasas de transmisión dependiendo si se trata del modo estándar, rápido o de alta velocidad, las cuales son 100 kbps, 400 kbps y 3.4 Mbps respectivamente. Este bus es muy utilizado en la industria, principalmente para comunicar microcontroladores con sus periféricos, propósito que también cumple en este proyecto. [22]

La transmisión de señales se realiza a través de dos conexiones denominadas SCL y SDA. La primera de ellas se usa para sincronizar la transferencia de datos a través del bus, haciendo uso de la señal de reloj del sistema, mientras que a través de la otra circulan los datos que los dispositivos intercambian. Todos los componentes que hagan uso del bus I²C deben estar conectados a a ambas líneas en topología tipo estrella. [22]

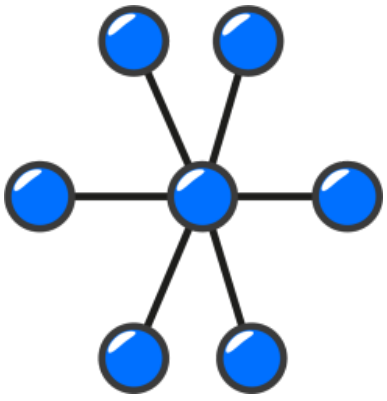

Figura 2.1 Topología tipo estrella [23].

Para que las comunicaciones funcionen de manera satisfactoria, es necesario que todos los dispositivos compartan una misma referencia a tierra. Tanto la línea SCL como la línea SDA son de tipo "drenador abierto" por lo que tienen un estado similar al de colector abierto pero asociado a un transistor FET. Debido a que se polarizan en estado alto, necesitan resistencias de *pull-up*. Esto permite una estructura de bus la cual

posibilita conectar en paralelo múltiples entradas y salidas. Al estar inactivas, ambas líneas se encontrarán en estado lógico alto. [22]

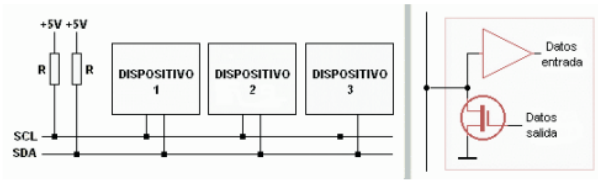

Figura 2.2 Conexiones del bus I²C con resistencias de pull-up y esquema de drenador abierto [22].

Cada dispositivo conectado al bus poseerá su propia dirección, de forma que se pueda usar esta para indicar a qué dispositivo afecta la transmisión estando varios de ellos conectados simultáneamente. Los dispositivos se dividen en maestros, los cuáles pueden iniciar una transmisión y gobiernan la línea de reloj, y esclavos que intervienen en la comunicación cuando un maestro se lo indica. En nuestro caso el microcontrolador actuará como maestro, mientras que los sensores tomarán el papel de esclavos. [22]

Cuando ambas señales estén en estado lógico alto, el bus se considera libre y un maestro podrá ocupar el bus estableciendo la condición de inicio, la cual consiste en un cambio de nivel alto a bajo de la línea SDA, mientras la línea SCL se mantiene a nivel alto, la única secuencia a parte de esta en la que se puede modificar la línea SDA manteniendo la línea SCL en alto es en la secuencia de parada, en la cual el cambio de nivel asociado a la línea SDA será inverso a la condición de inicio. [22]

Tras la condición de inicio, se transmitirá un byte que contendrá la dirección del dispositivo con el que se desea conectar y un indicador de si es una operación de lectura y escritura, si todo está correcto el esclavo responderá con un bit ACK que consistirá en un bit a nivel bajo tras el fin de la transmisión del maestro. Cada dispositivo comprueba si la dirección proporcionada es la suya, y si coincide enviará el bit ACK. La transmisión de datos comienzan por el MSB. [22]

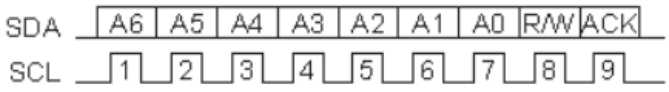

Figura 2.3 Inicio de transmisión en el protocolo I²C [22].

El bit de Lectura/Escritura tomará un nivel bajo para indicar escritura, en cuyo caso enviará bytes de datos tras comunicar en que registro desea escribir, recibiendo un bit ACK tras cada byte de información. Al terminar, llevará a cabo la secuencia de parada u otra secuencia de inicio si desea realizar una nueva transmisión. [22]

En caso de que se desee leer, el bit de Lectura/Escritura se encontrará en un nivel alto, por lo que el maestro sólo generará pulsos de reloj para que el esclavo pueda transmitir, siendo el maestro quien envíe los bits ACK en este caso. La operación de lectura es más compleja, pudiendo ser dividida en dos partes, una primera operación de escritura en la que se indica el registro interno a leer, y una segunda operación de lectura propiamente dicha en la que el esclavo transmite la información. [22]

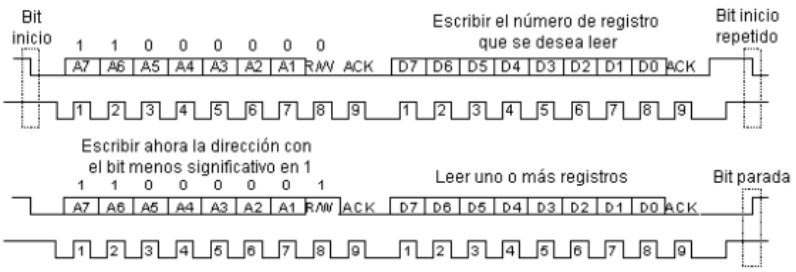

Figura 2.4 Secuencia de lectura en el protocolo I²C [22].

Durante la lectura, el esclavo gobierna la linea SDA, esto puede provocar que el maestro envíe pulsos de reloj sin que el esclavo pueda contestar debido a que no dispone del dato a leer de forma inmediata, provocando una lectura errónea. Para evitar que esto ocurra, el esclavo puede mantener la línea SCL a nivel bajo, retrasando la transmisión hasta disponer de la información requerida, esto es conocido como *clock stretching*. [22]

2.2 UART y Puerto Serie

La comunicación UART es uno de los protocolos más utilizados entre dispositivos para transmitir y recibir datos.

Al igual que en el bus I²C, el protocolo UART también usa sólo dos conexiones para entablar la transmisión, conocidas como TX y RX, las cuales forman el puerto serie. El funcionamiento de este protocolo es completamente diferente al explicado en la sección anterior, por definición, el protocolo UART posee una comunicación asíncrona en serie con velocidad configurable, lo cual significa que no existe señal de reloj para sincronizar los datos enviados desde un dispositivo a otro. [24]

El propósito del Pin TX es transmitir información, mientras que el pin RX es el encargado de recibirla, por lo tanto, para entablar la comunicación entre dos UART será necesario conectar sus pines TX al pin RX de la otra. Las UART además poseen un bus de datos que le manda la información de forma paralela, esta luego es transmitida de forma serial, bit a bit, hacia el receptor, para volver a ser convertida en datos en paralelo en el bus de datos asociado a la UART que ha recibido el byte. [24]

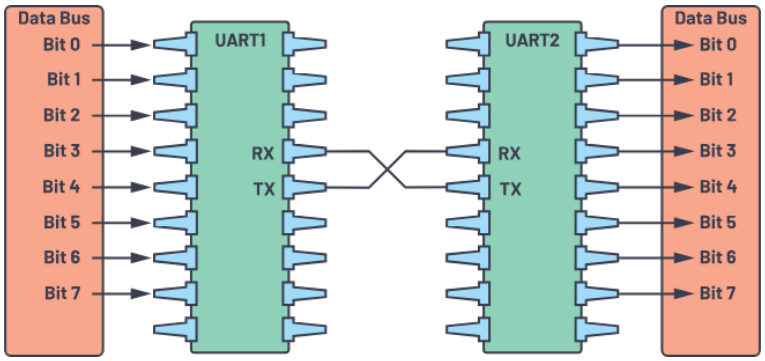

Figura 2.5 Conexionado de dos UART con buses de datos [24].

Para que la comunicación sea exitosa, ambos dispositivos deben trabajar a la misma velocidad, que puede ir desde los 9.600 hasta los 1.500.000 Bd, tomando valores predefinidos comprendidos entre estos límites, como puede ser 115.200 Bd, una de las velocidades más utilizadas. Al no existir señal de reloj en este protocolo, la sincronización se basa en que el emisor enviará los datos a la velocidad estipulada, mientras que el receptor usará esta misma para distinguir cada byte de información, haciendo cada uno uso de su reloj interno. La máxima diferencia de velocidad aceptable para que puedan sincronizarse es de un 10 %. [24]

Las transmisiones en el protocolo UART se realizan en forma de paquetes, que consisten de un bit de inicio, los datos a transmitir, un bit de paridad y uno o varios bits de parada.

Figura 2.6 Paquete UART [24].

Al igual que en el protocolo I²C, cuando la línea se encuentra libre se mantiene en un estado lógico alto, para comenzar la transmisión se mantendrá la línea en un estado lógico bajo durante un ciclo de reloj, cuando el receptor detecte este cambio empezará a leer a la tasa de baudios estipulada. A continuación, se enviará la información, que pueden ser de 5 a 8 bits si se usa bit de paridad, y hasta 9 en caso de que no se use. La transmisión suele comenzar por el LSB. El bit de paridad se utiliza para asegurar que la transmisión de datos fue correcta, ya que estos pueden variar debido a la radiación electromagnética, largas distancias en las transmisiones, o diferencias entre las tasas de baudios de los dos dispositivos, si el bit de paridad es un bit en

estado lógico bajo, el número total de bits en estado lógico alto en la transmisión debe ser par, por otro lado si el bit de paridad se encuentra en un estado lógico bajo, la suma de estos deben de ser impar. En caso de que el bit de paridad y la información proporcionada por los datos no coincida, la información se considera errónea. Por último, para terminar la transmisión se enviarán uno o dos cambios desde un estado lógico bajo a alto. [24]

En resumen, los pasos a seguir para realizar una transmisión UART son los siguientes:

1. La UART transmisora recibe datos en paralelo del bus de datos.
2. La UART transmisora añade a los datos el bit de inicio, el bit de paridad y el bit de parada.
3. La UART transmisora envía el paquete de forma serial a la UART receptora a la tasa de baudios prestablecida.

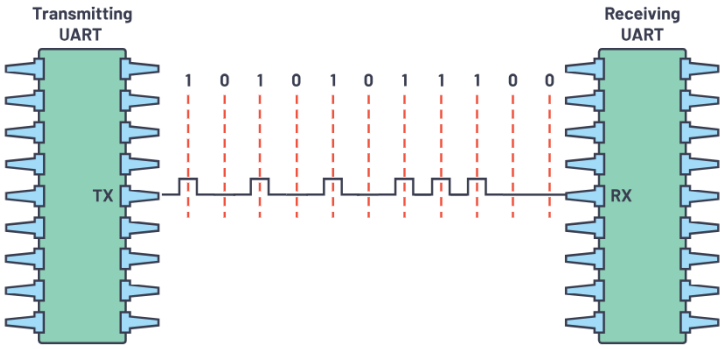

Figura 2.7 Transmisión UART [24].

4. La UART receptora elimina el bit de inicio, el bit de paridad, y el bit de parada.
5. La UART receptora transforma los datos de serie a paralelo y los envía a su bus de datos.

En este proyecto, el protocolo UART se utiliza para comunicar el microcontrolador con el ordenador y así poder obtener información durante la programación para el depurado del código, como el éxito a la hora de publicar en ThingSpeak, o los datos que proporcionan los sensores, por lo que el usuario final no los observará a menos que conecte el huerto automatizado a su ordenador en lugar de a una toma de corriente.

2.3 *Wi-Fi*

El Internet tal como lo conocemos se puede dividir en "capas", *Wi-Fi* es un estándar de comunicación extendido mundialmente que se engloba dentro de una de esas capas llamada capa de enlace, la cual se refiere a la unión física entre dos dispositivos, que en este caso se corresponde a una red que funciona en las bandas de frecuencia de 2.4 GHz o 5.2 GHz. [25]

En otras capas, como la capa de internet, podemos encontrar el protocolo IP, o el protocolo TCP en la capa de transporte, aunque no profundizaremos en estos. [25]

El microcontrolador que tenemos posee varios modos de configuración a la hora de trabajar con *Wi-Fi*. En el modo AP el ESP8266 generará su propia red *Wi-Fi*, a la que otros dispositivos puedan conectarse. [25]

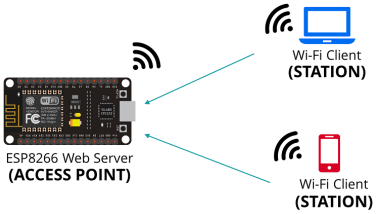

Figura 2.8 Esquema de conexiones al ESP8266 en modo AP [26].

El otro modo en que puede ser configurado, y el que se usará en este proyecto, es el modo ST, el cuál permite al ESP8266 conectarse a un router *Wi-Fi* que le de acceso a Internet. Tener acceso a Internet nos posibilitará subir los datos recogidos por los sensores a ThingSpeak en tiempo real, algo clave para el desarrollo del proyecto. [25]

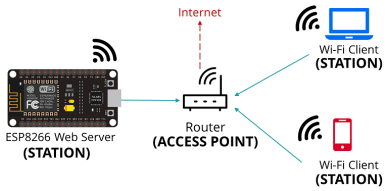

Figura 2.9 Esquema de conexiones dek ESP8266 en modo ST [26].

También existe la posibilidad de realizar ambas acciones de forma simultánea, aunque esta opción tampoco es aprovechada por nuestro sistema. [25]

Conectarse al router será la primera acción que realice nuestro microcontrolador al iniciarse, justo después de asegurar que los servomotores se encuentran detenidos, ya que estos pueden moverse de forma errática mientras se busca la señal, tras esto, se ejecutará el resto del programa.

Código 2.1 Código de Feather HUZZAH ESP8266 para realizar la conexión a un router *Wi-Fi*.

```
WiFi.begin(ssid, password);          // Nos conectamos a la red
Serial.print("Connecting to ");
Serial.print(ssid); Serial.println(" ...");

int i = 0;
while (WiFi.status() != WL_CONNECTED) { // Esperamos la conexion al wifi,
    reintentando cada segundo
delay(1000);
Serial.print(++i); Serial.print(' ');
 }

Serial.println('\n');
Serial.println("Connection established!");
Serial.print("IP address:\t");
Serial.println(WiFi.localIP());      // Manda por puerto serie la IP del ESP
    8266
```

3 Funcionamiento del Huerto Automatizado

El verdadero signo de la inteligencia no es el conocimiento sino la imaginación.

Albert Einstein

En este capítulo se comentará el comportamiento que tendrá el huerto automatizado durante su operación sobre el terreno, además de explicar el código que hace que actúe de tal manera.

3.1 Funcionamiento General

Lo más recomendable a la hora de regar una planta es hacerlo una vez al día y temprano en la mañana, ya que en esos instantes las temperaturas son más suaves y el agua se evaporará más lentamente, además, en estas horas las plantas inician su actividad metabólica, por lo que es un momento conveniente para proporcionarles nutrientes, vitaminas e hidratación.

Teniendo en mente estos hechos, configuraremos el sistema para que riegue a las 9 en punto de la mañana durante dos minutos, tiempo más que suficiente para que la bomba de agua se active, el agua alcance el final del tubo de riego e hidrate el terreno circular.

Para recorrer todo el terreno, la boca de riego será desplazada en movimientos circulares por la "grúa" que se encuentra en el centro del huerto. La grúa realizará giros en sentido horario y antihorario de forma alternativa, recogiendo o alargando el brazo, en cuya punta se encuentra la boca de riego cada vez que se complete una vuelta en uno de los sentidos, para así alcanzar tanto las zonas interiores como exteriores del huerto.

De forma simultánea, el sensor de temperatura, que también se encuentra en el extremo del brazo, irá recogiendo datos y publicándolos en ThingSpeak cada 20 segundos.

Cuando no sea hora de regar, el sensor recogerá los datos de un punto estático. Para tomar datos de otros puntos del huerto, cada vez que transcurra una hora la grúa hará al MLX90614 recorrer el huerto completo tres veces.

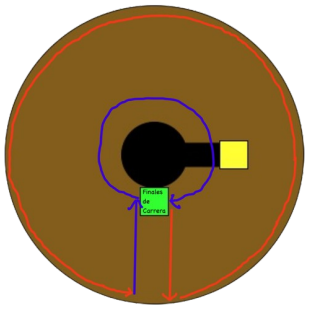

Figura 3.1 Boceto del Movimiento de la grúa.

A la vista de este planteamiento, es evidente que el código puede dividirse en dos partes, una máquina de estados que gobierne el movimiento y el riego, y otro bloque que se encargue de recoger los datos de los sensores y publicarlos en ThingSpeak.

3.2 Máquina de Estados: Movimiento y Riego

Una máquina de estados (o máquina de estados finitos) es una representación de un sistema reactivo basado en eventos que pasa de un estado a otro si se cumple la condición que controla el cambio. [27]

En nuestro caso, se utilizará para definir en qué momento comenzar a mover la grúa con los servomotores y qué movimiento realizar en cada ocasión, además, también se decidirá si activar la salida de riego o no.

El diagrama de nuestra máquina de estados es el siguiente:

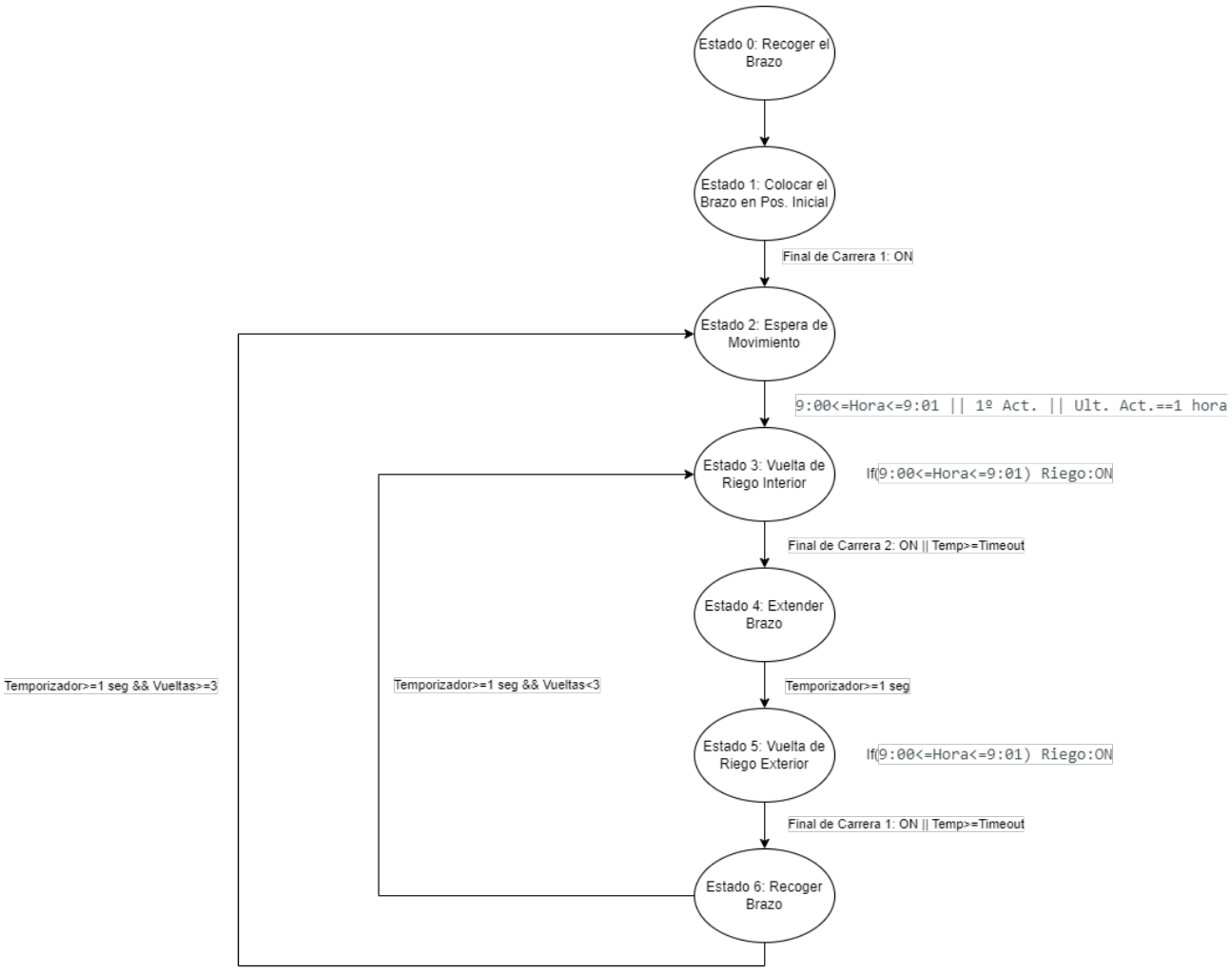

Figura 3.2 Máquina de estados de Movimiento y Riego.

A continuación se explicará el funcionamiento de cada estado, además de incluir el código que rige su comportamiento:

- Estado 0: Recoger el brazo.

  Este es el primer estado en el que entra el sistema cuando se conecta. Su única función es activar el servomotor que controla la longitud del brazo durante un segundo para recogerlo, de esta forma, cuando se inicie la secuencia de movimiento siempre empezará desde la misma posición. Al terminar el tiempo estipulado pasará directamente al siguiente estado.

Código 3.1 Código de Estado 0: Recoger el brazo.

```
///////Estado Inicial para recoger el brazo en caso de que se encuentre
    alargado////////

case 0:
servoEngranaje.write(120); //activamos el servomotor para recoger el brazo
delay(1000); //Detenemos un segundo con el motor girando
servoEngranaje.write(80);//parar servomotor del brazo
estado=1;
break;

//////////////////////////////////////////////////////////////////
```

- Estado 1: Colocar el brazo en Posición Inicial.

  Al igual que el estado anterior, su único objetivo es dirigir a la grúa central hacia la posición de inicio, en este caso se hará girar al servomotor de la base en sentido antihorario hasta que se active el final de carrera 1, de forma que esta termine en el límite derecho. Una vez que se cumpla la condición se pasará al siguiente estado.

Código 3.2 Código de Estado 1: Colocar el brazo en Posición Inicial.

```
////////////Estado Colocación de brazo en posición inicial////////////

case 1:

  servoBase.write(93);//giramos la base en sentido antihorario

  Pulsador=digitalRead(14);
  if (Pulsador==HIGH)
  {
  servoBase.write(76);
  estado=2;
  }
break;
//////////////////////////////////////////////////////////////////
```

- Estado 2: Espera de Movimiento.

  Este estado sirve para detener el funcionamiento del huerto cuando no sea necesario moverse, solo permitirá que la maquina de estados entre en el siguiente estado si se cumple una de las tres condiciones posibles:

  1. Es hora de regar (entre las 9:00 AM y las 9:02 AM).
  2. Ha pasado una hora desde la última vez que se activo.
  3. Es la primera vez que pasa por este estado tras conectarse.

Código 3.3 Código de Estado 2: Espera de Movimiento..

```
////////Estado para esperar a que sea la hora de regar, o que pase una
    hora, al iniciar se pasara directamente a moverse///////////////

case 2:
 if((now.hour()==9 && now.minute()>=0 && now.minute()<=1) || (horaact==-1
     && minact==-1) || (horaact<now.hour() && minact==now.minute()))
 {
  horaact=now.hour();
  minact=now.minute();
  vueltas=0;
  tiempoInicio = millis();
  estado=3;
 }
break;
//////////////////////////////////////////////////////////////////
```

- Estado 3: Vuelta de Riego Interior.

  Una vez pasado el estado 2, entramos en lo que podemos llamar "ciclo de movimiento", en este primer estado del ciclo se completará la vuelta interior. El microcontrolador ordenará al servomotor de la base girar en sentido horario hasta que se active el final de carrera 2 o haya un *timeout*, este tiempo límite se incluye debido a que en ciertas ocasiones los servomotores pueden tardar más tiempo del debido en pulsar los finales de carrera, quedando atorados temporalmente intentando que estos se cierren, el *timeout* evita que esto suceda.

  En este estado además se comprobará si se debe activar o no la bomba de agua para regar y se aumentarán la cuenta de vueltas en 1.

Código 3.4 Código de Estado 3: Vuelta de Riego Interior.

```
////////Estado Vuelta de riego Interior////////////////////

case 3:
    //Si es hora de regar, activar la bomba
  if((now.hour()==9 && now.minute()>=0 && now.minute()<=1))
  digitalWrite(12, HIGH);
  else digitalWrite(12, LOW);
  servoBase.write(65);//giramos la base en sentido horario

  //cuando se pulse el final de carrera. paramos y preparamos para recoger
       el brazo

  Pulsador=digitalRead(16);
  if (Pulsador==HIGH || (millis() - tiempoInicio) > timeoutgiro) //Detener
       si se pulsa el final de carrera o pasa el timeout
  {
  servoBase.write(76);
  vueltas++;
  tiempoInicio = millis();
  estado=4;
  }
break;
//////////////////////////////////////////////////////////////////
```

- Estado 4: Extender el brazo.

  Este estado es similar al estado 1, su única función es extender el brazo para poder comenzar la vuelta de riego exterior, tras transcurrir un segundo, que es el tiempo que el servomotor tarda en extender el brazo, se pasará al siguiente estado.

Código 3.5 Código de Estado 4: Extender el brazo.

```
///////Estado Extender brazo//////////////////////////

case 4:
  if ((millis() - tiempoInicio) < tiempoBrazo) { //girar mientras estemos
      dentro de tiempo
  servoEngranaje.write(40);
  }
  else{
   servoEngranaje.write(80);
   tiempoInicio = millis();
   estado=5;
  }
break;
//////////////////////////////////////////////////////////////////
```

- Estado 5: Vuelta de Riego Exterior.

  Este caso es exactamente igual que el estado 3, con la única diferencia de que el giro es en sentido antihorario y se atiende a la activación del final de carrera 1.

Código 3.6 Código de Estado 5: Vuelta de Riego Exterior.

```
////////////////Estado Vuelta de Riego Exterior/////////////////

case 5:
  //Si es hora de regar, activar la bomba
  if((now.hour()==9 && now.minute()>=0 && now.minute()<=1))
  digitalWrite(12, HIGH);
  else digitalWrite(12, LOW);
  servoBase.write(93);//giramos la base en sentido antihorario

  //cuando se pulse el final de carrera. paramos y preparamos para alargar
       el brazo
  Pulsador=digitalRead(14);
  if (Pulsador==HIGH || (millis() - tiempoInicio) > timeoutgiro) //Detener
       si se pulsa el final de carrera o pasa el timeout
  {
  servoBase.write(76);
  tiempoInicio = millis();
  estado=6;
  }
break;
//////////////////////////////////////////////////////////////////
```

- Estado 6: Recoger el brazo.

  Esta es la última fase del ciclo de movimiento, su funcionamiento es totalmente igual al estado 4, con la diferencia de que el giro se realiza en sentido contrario.

Cuando este estado finalice se comprobará si se han completado 3 ciclos, en caso afirmativo, se pasará al estado 2 para esperar a que sea de nuevo momento de moverse, en caso contrario, saltará al estado 3 para completar las vueltas restantes.

Código 3.7 Código de Estado 6: Recoger el brazo.

```
//////////////////Estado Recoger brazo//////////////////////////

case 6:

  if (((millis() - tiempoInicio) < tiempoBrazo)) { //girar mientras
      estemos dentro de tiempo
  servoEngranaje.write(120);
  }
  else{
   servoEngranaje.write(80);
   if(vueltas>=3) estado=2; //comprobar si se han dado 3 vueltas
   else estado=3;
  }
break;
//////////////////////////////////////////////////////////////////////
```

3.3 Recogida de Datos y Publicación en ThingSpeak

A parte de la máquina de estados, el código cuenta, como ya hemos comentado, con otra sección que será la encargada de gestionar la obtención de datos por parte del sensor de temperatura y publicar tal información tanto en ThingSpeak como en el puerto serie para compararla con la existente en el canal.

Esta sección se activará cada 20 segundos, ya que ThingSpeak solo permite publicar un nuevo dato cada 15 segundos. 5 segundos extra han sido añadidos para asegurar que no se provocan errores en la subida debido a no respetar el espaciado temporal entre datos.

Una vez que se active, el código actualizará el tiempo correspondiente a la última vez que publicó, ordenará al sensor que tome la temperatura del terreno y publicará esta información en el canal asignado en ThingSpeak. Tras esto, indicará por puerto serie si pudo publicar correctamente, junto con la temperatura medida y la hora que proporciona el RTC.

Código 3.8 Código de Recogida de Datos y Publicación en ThingSpeak.

```
//////////Subida de datos a ThingSpeak/////////////////////////
// Comprobar si se ha reiniciado el temporizador
if (millis() < ultimoTiempo)
{
// Asignar un nuevo valor
ultimoTiempo = millis();
tiempoInicio= millis();
}
if ((millis() - ultimoTiempo) > intervaloEnvio)
{
// Marca de tiempo para el siguiente intervalo
ultimoTiempo = millis();

// Obtener temperatura Objeto grados Celsius
float temperaturaCampo = termometroIR.readObjectTempC();
```

```
//Definimos que en el campo 1 vamos a subir la variable temperaturaCampo
ThingSpeak.setField(1,temperaturaCampo);
//Subimos los datos al canal de Thingspeak
int x = ThingSpeak.writeFields(ChannelNumber, WriteAPIKey);
if (x == 200) { //En caso de exito
  Serial.println("Canal actualizado.");
  }
else { //Si falla
  Serial.println("Problema actualizando Canal, HTTP error code " + String(x));
  }

  //Mostramos info de RTC y sensor por puerto Serie
 Serial.print(now.day());
 Serial.print('/');
 Serial.print(now.month());
 Serial.print('/');
 Serial.print(now.year());
 Serial.print(" ");
 Serial.print(now.hour());
 Serial.print(':');
 Serial.print(now.minute());
 Serial.print(':');
 Serial.print(now.second());
 Serial.println();
 Serial.print("Temp. Terreno => ");
 Serial.print(temperaturaCampo);
 Serial.println("ºC");
}
//////////////////////////////////////////////////////////////////////
```

4 Construcción del Huerto Automatizado

Los científicos estudian el mundo tal como es; los ingenieros crean el mundo que nunca ha sido.

Theodore von Kármán

En este capítulo se explicará cómo se conectan los diferentes elementos electrónicos pertenecientes al sistema y cómo se construyó la estructura que conforma el huerto automatizado.

4.1 Conexionado

El núcleo de nuestro sistema es el microcontrolador ESP8266, prácticamente todos los otros dispositivos se encuentran conectados a él de alguna forma, además, este se encarga de alimentar a la amplia mayoría de ellos con una de sus salidas de voltaje. El esquema eléctrico es el siguiente:

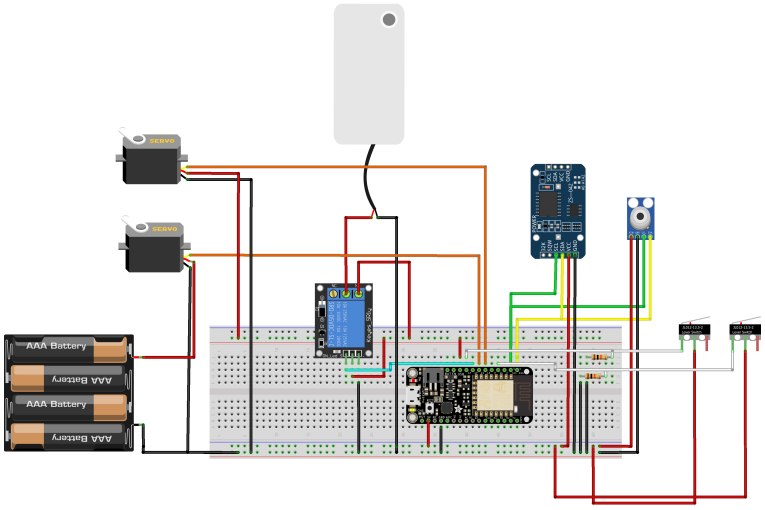

Figura 4.1 Esquema eléctrico del huerto automático.

- Tanto el sensor MLX90614 como el RTC se conectan en sus puertos SCL y SDA a los pines 4 y 5 respectivamente, los cuáles se corresponden a la comunicación a través del bus I²C. Adicionalmente, será necesario conectarlos a tierra y al pin de 3.3 V para su alimentación.

- El relé se conectará al pin de 5 V y a tierra, además de al pin 12, que estará configurado como salida, para recibir la señal de cierre cuando sea necesario. A su vez, su entrada NA también se conectará a 5 V, mientras que en su entrada común encontraremos el pin positivo de la bomba a ser controlada, cuyo pin negativo se encuentra conectado a tierra.

- Los finales de carrera 1 y 2 se conectarán en sus entradas NA a los pines 14 y 16, respectivamente, mientras que su entrada común irá al pin de 3.3 V, ya que los pines del ESP8266 no están preparados para recibir señales de 5 V. A estos mismos pines es necesario conectar unas resistencias de *pull-down* , de forma que cuando los finales de carrera se encuentren abiertos, las entradas del microcontrolador registren un valor lógico bajo.

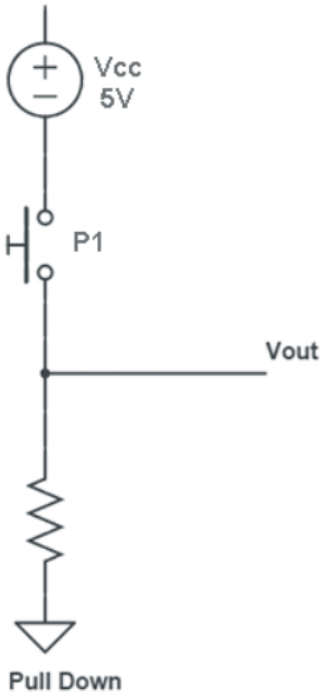

Figura 4.2 Esquema Eléctrico de una resistencia de *pull-down* [28] .

- El servomotor que se encuentra en la base de la estructura se controla a través de una señal PWM mandada desde el pin 13, mientras que el pin 15 se encarga de controlar al servomotor adherido al engranaje que extiende o recoge el brazo de la grúa. Ambos se encuentran conectados a tierra, sin embargo, mientras que el servomotor del engranaje recibe corriente desde el pin de 5 V, el de la base se alimenta desde una batería externa de 6 V formada por 4 pilas de 1.5 V en serie para obtener algo de fuerza extra a la hora de realizar sus movimientos.

4.2 Estructura del Huerto Automático

La estructura del huerto automático tiene varias partes bien diferenciadas, una base que como ya comentamos en la sección del diseño de concepto, actúa como cobijo para los componentes electrónicos y el depósito de agua, y por otro lado encontraremos al terreno circular junto con la grúa que actúa sobre él.

4.2.1 Base del Huerto Automático

La base está formada por una caja de corcho para que sea ligera y fácil de transportar, además, este material absorbe el agua, protegiendo a los componentes electrónicos en caso de una fuga, situación para la que también posee aberturas en la parte inferior con el objetivo de evacuar el líquido . En el exterior de esta encontraremos un portapilas en uno de los laterales, situado aquí para facilitar el cambio de baterías y la desconexión de estas del circuito si fuera necesario, gracias a un interruptor, que también puede ejercer como medida de seguridad. También es posible localizar el cable USB que alimenta al sistema patiendo desde aquí.

En su interior podremos observar el depósito que aloja a la bomba de agua, colocado con cierta inclinación hacia donde se encuentra esta, de forma que pueda usar toda el agua que esté disponible. El depósito cuenta

con una abertura en la parte superior por la que se puede llenar, además de otra salida en uno de sus laterales desde el que el tubo de riego sube hasta la grúa.

Por otro lado, también se encuentra aquí la protoboard que acoge a la amplia mayoría de elementos electrónicos, rodeada de una pequeña caja protectora, cuya base también es de corcho para absorber agua. Desde aquí partirán cables por el interior del tubo que sostiene la grúa y por otro tubo colocado en uno de los laterales hacia los componentes que forman la grúa y así poder conectarlos con los que se encuentran en el interior.

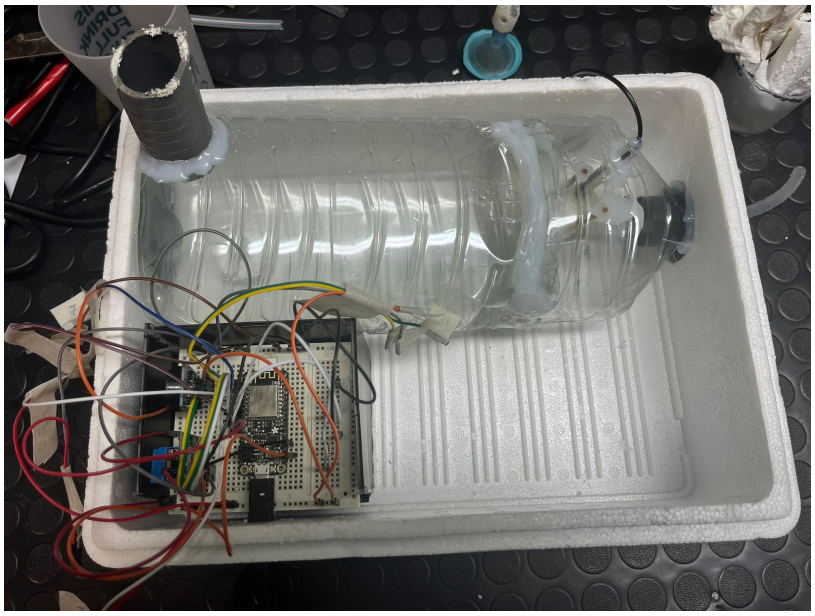

Figura 4.3 Interior de la base del Huerto Automático .

4.2.2 Terreno Circular

Pasaremos ahora a describir la zona donde se encontrarán las plantas.

Sobre la base se encuentra una zona circular recubierta de plástico para evitar que el agua se escape, sobre este plástico se colocará la tierra, que no será demasiado profunda ya que solo busca ser ilustrativa y capaz de acoger a algunas plantas pequeñas. En el borde de la base encontramos un recubrimiento curvado con la idea de que el agua que caiga sobre ella fluya hacia el terreno, ya que la grua en su máxima extensión puede derramar cierta cantidad de líquido sobre él.

En el centro del terreno se coloca el tubo sobre el que se encuentra situado la grúa, junto a otra barra que sostiene los finales de carrera en su lugar. Ambos están rodeados por una pequeña barrera con una tapa que evita que el agua o la tierra alcancen el interior de la base a través de las aberturas para el cableado.

Figura 4.4 Terreno del Huerto Automático .

4.2.3 Grúa

Sobre el tubo situado en el centro del terreno circular encontramos el primer servomotor, artífice del giro de la grúa y el que sostiene al resto de elementos. Adherido a él tenemos una cremallera costruida con piezas de lego, que gracias a su ligereza y gran resistencia son ideales para este proyecto. Lego también fabrica engranajes, en concreto nosotros usaremos uno de 12 dientes, que accionado por el otro servomotor, el cuál se encuentra protegido por una pequeña caja de plástico, hará avanzar o retroceder la cremallera, moviendo el brazo de la grúa. Cabe destacar que esta cremallera puede alcanzar una extensión mayor, pero debido al limitado tamaño del terreno ha sido inevitable restringir su movimiento para que no saliera agua fuera del espacio restringido por la barrera.

En la parte trasera de la grúa se ha colocado un contrapeso para equilibrar la carga que sufre la parte delantera de la estructura, donde además de uno de los servomotores encontramos al sensor de temperatura, también protegido por una caja similar a la del servomotor, apuntando hacia abajo. Este se encuentra acompañado por la salida de riego, la cuál está unida a un pequeño mecanismo extraído de un bote de jabón que le permite girar y así facilitar el movimiento. Hasta estos elementos llegan el tubo de riego y los cables que parten desde la base atravesando el tubo lateral. Como llegan a esta posición desde arriba, permiten que la grúa gire sin enredarse en ellos.

Por último, en la parte inferior de la cremallera se localizan, dos pequeñas barras de metal rematadas por tuercas. La función de estas es golpear a los finales de carrera a su paso, o en el caso de no poder pulsarlos, retener a la grúa, para dar por finalizado el movimiento en la dirección que se esté ejecutando en ese momento.

Figura 4.5 Grúa del Huerto Automático .

5 Resultados y Conclusiones

Para tener éxito, la planificación sola es insuficiente. Uno debe improvisar también.

Isaac Asimov

En este capítulo se reflexionará sobre los logros conseguidos durante el desarrollo del proyecto, además de abordar las posibles líneas de mejora disponibles como los problemas encontrados durante el desarrollo del mismo.

5.1 Resultados y conclusiones

El sistema descrito en estas páginas ha conseguido satisfacer los objetivos propuestos al inicio del desarrollo, siendo un ejemplo de huerto que puede realizar funciones básicas de manera autónoma y proporcionar información pertinente a su usuario de forma remota, sin tener un costo excesivamente elevado.

Aún así, el sistema es sólo un prototipo , que aunque estaría listo para funcionar con plantas de pequeña escala, en caso de ser utilizado con plantas de mayor tamaño, como tomates o pimientos, rwqueriría de diversos cambios para ser adecuado, como el uso de motores más potentes y una estructura más resistente y elaborada. De igual manera, estos objetivos salen del alcance de este trabajo y se dejarán plateados para posibles revisiones futuras.

También cabe resaltar, que aunque el huerto es funcional, el aprovechamiento del terreno podría haberse realizado de manera más eficiente, ya que el uso de finales de carrera limita el movimiento y nos hace prescindir de una zona de riego, igualmente, la extensión de la grúa habría permitido alcanzar mayores distancias, pero la limitación en los recursos disponibles no ha hecho posible explotar al máximo estas cualidades.

5.2 Problemas encontrados durante el desarrollo

Durante la ejecución del proyecto se han enfrentado numerosos problemas que se recogerán a continuación, los cuáles han sido solventados de la mejor manera que nuestras capacidades nos han permitido:

- La grúa no podía girar siempre en el mismo sentido, ya que esto produciría que se enreden los cables a su alrededor, por lo que ha sido necesario ir alternando direcciones.
- Los servomotores necesitaban algún tipo de realimentación para saber cuando se había completado una vuelta, ya que el uso de un temporizador no era lo suficientemente preciso y provocaría que la grúa acabe desplazandose hasta enredar los cables. La única solución posible fue hacer uso de finales de carrera, sacrificando una parte del terreno.
- Los cables provenientes desde la base hacia la punta de la grúa debían llegar desde arriba para no enredarse en la columna que actúa como base, por lo que fue necesario colocar un tubo exterior de mayor altura para guiarlos de forma correcta.
- Una vez terminado el proyecto, hubo una fuga de agua en el depósito que ha obligado a sustituirlo y reemplazar la caja que actuaba como base.

5.3 Líneas de mejora

Para finalizar, hablaremos de las posibles opciones disponibles de cara a aumentar las capacidades o mejorar el funcionamiento del sistema en futuras versiones:

- Instalar sondas repartidas por el terreno para controlar otras variables, como pueden ser la humedad o la salinidad.
- Hacer uso de un sensor de nivel para conocer el agua disponible en el depósito en todo momento.
- Sustituir los servomotores por motores paso a paso, que debido a su mayor precio y coste no han sido usados para esta versión, lo que permitiría eliminar los finales de carrera y mejorar la precisión de los movimientos.
- Mejorar la calidad de la estructura para hacerla más resistente.

6 Presupuesto

Concepto	Unidades	Pedido mínimo	Precio Unitario (con IVA)	Total (con IVA)
Microcontrolador Feather Huzzah ESP8266	1	1	17,90 €	17,90 €
Sensor de Temperatura MLX90614	1	1	18,99 €	18,99 €
Bomba de Agua	1	1	8,11 €	8,11 €
Relé de 5V	1	1	2,39 €	2,39 €
Cables (Conjunto de 120)	2	1	6,99 €	13,98 €
Finales de Carrera	2	10	0,599 €	5,99 €
Resistencias	2	60	0,1165 €	6,99 €
Servomotor 360º	2	1	5,60 €	11,20 €
Portapilas	1	1	6,75 €	6,75 €
RTC DS3231	1	1	6,99 €	6,99 €
Protoboard	1	3	3,663 €	10,99 €
Piezas Lego	-	-	-	5,58 €
Otros materiales	-	-	-	20 €
				135,86 €

7 Anexo

Código 7.1 Código Completo del Huerto Automático.

```
  #include <Wire.h>
#include <RTClib.h> //Libreria RTC
#include "ThingSpeak.h" //Libreria comandos Thingspeak
#include <Adafruit_MLX90614.h> //Libreria sensor de temperatura infrarrojo
#include <ESP8266WiFi.h>     //Libreria Wifi
#include <Servo.h> //libreria Servos

//variables para controlar la maquina de estados
int estado=0,vueltas=0;
int Pulsador=0; //variable estado finales de carrera
DateTime now; //variable hora actual
int horaact=-1,minact=-1; //variables ultima activacion

//Variables para controlar los servos
Servo servoBase, servoEngranaje; //servoEngranaje cruz servoBase palo
int tiempoBrazo = 1000;//tiempo que tarda en alargar el brazo(ajustar con todo
    conectado)
int timeoutgiro = 4000;//tiempo para girar hacia el otro lado si el brazo se
    atasca
unsigned long tiempoInicio; //variables de tiempo para determinar cuando se
    inicia el movimiento

//const char* ssid   = "DIGIFIBRA-sdZ6";      //El SSID (nombre) del wifi al
    que vamos a conectarnos (casa)
//const char* password = "P3bueuAkQNRd"; // Contraseña del wifi (casa)

const char* ssid   = "Jaime";       // El SSID (nombre) del wifi al que vamos a
     conectarnos (movil)
const char* password = "calleolivares21"; //Contraseña del wifi (movil)

WiFiClient client;

//Datos acceso Canal ThingSpeak
unsigned long ChannelNumber = 2439693;
const char * WriteAPIKey = "1HHYTQP3FEUAGVKK";

//Temporizador ThingSpeak 20 seg
```

```
unsigned long ultimoTiempo = 0; // almacena la ultima vez que se lanzo nuestro
    evento
unsigned long intervaloEnvio = 20000; // 20 segundos

// Declaramos un RTC DS3231
RTC_DS3231 rtc;
//Declaramos un MLX90614
Adafruit_MLX90614 termometroIR = Adafruit_MLX90614();

void setup () {
 Serial.begin(115200); //Iniciar comunicacion por puerto serie a 115200 baudios
 delay(10);
 Serial.println('\n');

servoBase.attach(13); //Pin Servo de la Base 13
servoEngranaje.attach(15); //Pin Servo del brazo 15
servoBase.write(76); //Parar los servos
servoEngranaje.write(80);

 WiFi.begin(ssid, password);         // Nos conectamos a la red
 Serial.print("Connecting to ");
 Serial.print(ssid); Serial.println(" ...");

 int i = 0;
 while (WiFi.status() != WL_CONNECTED) { // Esperamos la conexion al wifi,
    reintentando cada segundo
 delay(1000);
 Serial.print(++i); Serial.print(' ');
  }

 Serial.println('\n');
 Serial.println("Connection established!");
 Serial.print("IP address:\t");
 Serial.println(WiFi.localIP());     // Manda por puerto serie la IP del ESP
    8266

// Iniciar termómetro infrarrojo con Arduino
  termometroIR.begin();

 delay(3000);

 // Comprobamos si tenemos el RTC conectado
if (! rtc.begin()) {
 Serial.println("No hay un módulo RTC");
 while (1);
 }

 // Ponemos en hora, solo la primera vez, luego comentar y volver a cargar.
 // Ponemos en hora con los valores de la fecha y la hora en que el sketch ha
    sido compilado.
 //rtc.adjust(DateTime(F(__DATE__), F(__TIME__)));

ThingSpeak.begin(client); // Inicializamos ThingSpeak

pinMode(12, OUTPUT);//Pin del relé para la bomba de agua 12
pinMode(14, INPUT); //Pin del final de carrera 1
pinMode(16, INPUT); //Pin del final de carrera 2
```

```
}

void loop () {

DateTime now = rtc.now(); //Obtener hora del RTC

// Comprobar si se ha reiniciado el temporizador
if (millis() < tiempoInicio)
{
// Asignar un nuevo valor
tiempoInicio = millis();
}

switch(estado){ //Maquina de estados del huerto

////Estado Inicial para recoger el brazo en caso de que se encuentre alargado
    ////
case 0:
servoEngranaje.write(120); //activamos el servomotor para recoger el brazo
delay(1000); //Detenemos un segundo con el motor girando
servoEngranaje.write(80);//parar servomotor del brazo
estado=1;
break;
//////////////////////////////////////////////////


/////////Estado Colocación de brazo en posición inicial///////////
case 1:

  servoBase.write(93);//giramos la base en sentido antihorario

  Pulsador=digitalRead(14);
  if (Pulsador==HIGH)
  {
  servoBase.write(76);
  estado=2;
  }
break;
//////////////////////////////////////////////////////////////////


////////////Estado para esperar a que sea la hora de regar, o que pase una hora
    , al iniciar se pasara directamente a moverse//////////
case 2:
 if((now.hour()==9 && now.minute()>=0 && now.minute()<=1) || (horaact==-1 &&
     minact==-1) || (horaact<now.hour() && minact==now.minute()))
 {
  horaact=now.hour();
  minact=now.minute();
  vueltas=0;
  tiempoInicio = millis();
  estado=3;
 }
break;
//////////////////////////////////////////////////////
```

```
////////////Estado Vuelta de riego Interior////////////////////////
case 3:
    //Si es hora de regar, activar la bomba
  if((now.hour()==9 && now.minute()>=0 && now.minute()<=1)) digitalWrite(12,
      HIGH);
  else digitalWrite(12, LOW);
  servoBase.write(65);//giramos la base en sentido horario

  //cuando se pulse el final de carrera. paramos y preparamos para recoger el
      brazo
  Pulsador=digitalRead(16);
  if (Pulsador==HIGH || (millis() - tiempoInicio) > timeoutgiro) //Detener si
      se pulsa el final de carrera o pasa el timeout
  {
  servoBase.write(76);
  vueltas++;
  tiempoInicio = millis();
  estado=4;
  }
break;
//////////////////////////////////////////////////////


//////////////Estado Extender brazo////////////////////////////////
case 4:
  if ((millis() - tiempoInicio) < tiempoBrazo) { //girar mientras estemos
      dentro de tiempo
  servoEngranaje.write(40);
  }
  else{
   servoEngranaje.write(80);
   tiempoInicio = millis();
   estado=5;
  }
break;
//////////////////////////////////////////////////


//////////////Estado Vuelta de Riego Exterior////////////////////
case 5:
  //Si es hora de regar, activar la bomba
  if((now.hour()==9 && now.minute()>=0 && now.minute()<=1)) digitalWrite(12,
      HIGH);
  else digitalWrite(12, LOW);
  servoBase.write(93);//giramos la base en sentido antihorario

  //cuando se pulse el final de carrera. paramos y preparamos para alargar el
      brazo
  Pulsador=digitalRead(14);
  if (Pulsador==HIGH || (millis() - tiempoInicio) > timeoutgiro) //Detener si
      se pulsa el final de carrera o pasa el timeout
  {
  servoBase.write(76);
  tiempoInicio = millis();
  estado=6;
  }
break;
```

```
////////////////////////////////////////////////////////////


//////////////////Estado Recoger brazo//////////////////////
case 6:

  if (((millis() - tiempoInicio) < tiempoBrazo)) { //girar mientras estemos
      dentro de tiempo
  servoEngranaje.write(120);
  }
  else{
   servoEngranaje.write(80);
   if(vueltas>=3) estado=2; //comprobar si se han dado 3 vueltas
   else estado=3;
  }
break;
////////////////////////////////////////////////////////////
}

///////////Subida de datos a ThingSpeak//////////////////////
// Comprobar si se ha reiniciado el temporizador
if (millis() < ultimoTiempo)
{
// Asignar un nuevo valor
ultimoTiempo = millis();
tiempoInicio= millis();
}
if ((millis() - ultimoTiempo) > intervaloEnvio)
{
// Marca de tiempo para el siguiente intervalo
ultimoTiempo = millis();

// Obtener temperatura Objeto grados Celsius
float temperaturaCampo = termometroIR.readObjectTempC();

//Definimos que en el campo 1 vamos a subir la variable temperaturaCampo
ThingSpeak.setField(1,temperaturaCampo);
//Subimos los datos al canal de Thingspeak
int x = ThingSpeak.writeFields(ChannelNumber, WriteAPIKey);
if (x == 200) { //En caso de exito
  Serial.println("Canal actualizado.");
  }
else { //Si falla
  Serial.println("Problema actualizando Canal, HTTP error code " + String(x));
  }

  //Mostramos info de RTC y sensor por puerto Serie
 Serial.print(now.day());
 Serial.print('/');
 Serial.print(now.month());
 Serial.print('/');
 Serial.print(now.year());
 Serial.print(" ");
 Serial.print(now.hour());
 Serial.print(':');
 Serial.print(now.minute());
 Serial.print(':');
```

```
 Serial.print(now.second());
 Serial.println();
 Serial.print("Temp. Terreno => ");
 Serial.print(temperaturaCampo);
 Serial.println("ºC");
}
//////////////////////////////////////////////////////////////////

}
```

Índice de Figuras

Índice de Tablas

Índice de Códigos

Bibliografía

[1] N. Khan, R. L. Ray, G. R. Sargani, M. Ihtisham, M. Khayyam, and S. Ismail, "Current progress and future prospects of agriculture technology: Gateway to sustainable agriculture," *Sustainability*, vol. 13, no. 9, 2021.

[2] E. H. lady ada and B. Rubell, "Adafruit Feather HUZZAH ESP8266," https://learn.adafruit.com/adafruit-feather-huzzah-esp8266, accessed: February 2024.

[3] S. Madakam, R. Ramaswamy, and S. Tripathi, "Internet of things (iot): A literature review," *Journal of Computer and Communications*, vol. 3, no. 5, pp. 164–173, 2015.

[4] *Analog Continuous Rotation Servo Motor, 6V, 360°*, Multicomp Pro, 3 2020, rev 1.

[5] B. de Bakker, "How to control a 360 degree servo motor with Arduino," https://www.makerguides.com/es/how-to-control-a-360-degree-servo-motor-with-arduino/#What_makes_a_continuous_servo_motor_special, accessed: February 2024.

[6] *MLX90614 family*, Melexis, 9 2019, rev 13.

[7] L. del Valle Hernández, "Termómetro infrarrojo con Arduino MLX90614," https://programarfacil.com/blog/arduino-blog/termometro-infrarrojo-con-arduino-mlx90614/#Caracteristicas_tecnicas_del_MLX90614, accessed: February 2024.

[8] ——, "Reloj con Arduino, cómo controlar los tiempos con un RTC," https://programarfacil.com/blog/arduino-blog/reloj-con-arduino-rtc/, accessed: February 2024.

[9] *Extremely Accurate I2C-Integrated RTC/TCXO/Crysta*, Dallas Semiconductor, 3 2010, rev 7.

[10] ADSLZone, "Qué tipos de pilas de botón hay y en qué se diferencian," https://www.adslzone.net/reportajes/tecnologia/tipos-pilas-boton/, accessed: April 2024.

[11] M. Electrónica, "Mini Bomba De Agua Sumergible De 3-6v Blanca 120l/h," https://mvelectronica.com/producto/mini-bomba-de-agua-sumergible-de-3-6v-blanca-120l-h, accessed: Abril 2024.

[12] Ardest, "Módulo de relé de canal de 5 V," https://www.amazon.com/-/es/Ardest-Arduino-Raspberry-expansi%C3%B3n-indicador/dp/B074JJ9B7Z3, accessed: February 2024.

[13] Voltione, "¿Qué es un relé y para qué sirve?" https://voltione.com/pages/que-es-rele, accessed: Abril 2024.

[14] G. Eina, "Elemento de control: relé," https://buscadordetalleres.com/blog/elemento-de-control-rele/, accessed: Abril 2024.

[15] Cetronic, "PLACA PROTOBOARD 165x55x10mm," https://www.cetronic.es/sqlcommerce/disenos/plantilla1/seccion/producto/DetalleProducto.jsp?idIdioma=1&idTienda=93&codProducto=999441012&cPath=1017, accessed: Abril 2024.

[16] Electronicaplugandplay, "Resistencia 10 Kohm 1/4 Watt," https://www.electronicaplugandplay.com/componentes-pasivos/product/97-resistencia-10k-1-4-watt, accessed: Abril 2024.

[17] Robotlandia, “Final de carrera KW11-N interruptor de rodillo End-Stop,” https://robotlandia.es/movimiento-y-distancia/748-final-de-carrera-kw11-n-interruptor-de-rodillo-end-stop.html, accessed: Abril 2024.

[18] Rebower, “Rebower Protoboard Cinta de Cable,” https://www.amazon.es/Rebower-Protoboard-Puente-Bricolaje-Experimento/dp/B0BNKMG223, accessed: February 2024.

[19] Cablematic, “Portapilas plano para 4 pilas LR6 AA 1.5V,” https://cablematic.com/es/productos/portapilas-plano-para-4-pilas-lr6-aa-15v-EN090/, accessed: April 2024.

[20] Arduino, “Overview of the Arduino IDE 1,” https://docs.arduino.cc/software/ide-v1/tutorials/Environment/, accessed: Abril 2024.

[21] Mathworks, “ThingSpeak for IoT,” https://thingspeak.com/pages/commercial_learn_more, accessed: Abril 2024.

[22] V. M. Ayora, “Análisis y control de un vehículo basado en péndulo invertido,” 2009.

[23] D. C. Forum, “Topología de los sistemas,” https://domoticautem.wordpress.com/topologia-de-los-sistemas/l, accessed: April 2024.

[24] E. Peña and M. G. Legaspi, “UART: A Hardware Communication Protocol Understanding Universal Asynchronous Receiver/Transmitter,” https://www.analog.com/en/resources/analog-dialogue/articles/uart-a-hardware-communication-protocol.html#author, accessed: April 2024.

[25] P. P, “A Beginner’s Guide to the ESP8266,” https://tttapa.github.io/ESP8266/Chap05%20-%20Network%20Protocols.html, accessed: February 2024.

[26] R. N. T. C. Forum, “ESP8266 NodeMCU Access Point (AP) for Web Server,” andomnerdtutorials.com/esp8266-nodemcu-access-point-ap-web-server/l, accessed: April 2024.

[27] MathWorks, “Introducción a las state machines,” https://es.mathworks.com/discovery/state-machine.html, accessed: April 2024.

[28] L. del Valle Hernández, “Resistencia pull up y pull down,” https://programarfacil.com/blog/arduino-blog/resistencia-pull-up-y-pull-down/, accessed: April 2024.

Glosario

ACK *Acknowledgement*. 14

AP *Access Point*. 16, 41

Bd Baudio. 15

EEPROM *Electrically Erasable Programmable Read-Only Memory*. 7

FET *Field-effect transistor*. 13

FOV *Field of View*. 6, 41

GHz GigaHerzio. 16

GPIO *General Purpose Input/Output*. 3

Hz Herzio. 4

IDE *Integrated Development Environment*. 10, 11, 41, VII

IoT *Internet of Things*. 3, 11

IP *Internet Protocol*. 3, 16

IR Infrarroja. 5

I²C *Inter-Integrated Circuit*. 3, 6, 7, 11, 13–15, 27, 41, VII

KΩ Kilohmio. 9, 41

kbps Kilobyte por segundo. 13

Kg-cm Kilogramo-centímetro. 4

kHz kiloHerzio. 7

L/h Litros por hora. 8

LSB *Least Significant Bit*. 15

m Metro. 8

mA MiliAmperio. 4, 8

MB MegaByte. 3

Mbps Megabyte por segundo. 13

MHz MegaHerzio. 3

ms Milisegundo. 4

MSB *Most Significant Bit*. 14

NA Normalmente abierto. 8, 27, 28

NC Normalmente cerrado. 8

PWM *Pulse Width Modification*. 4, 28, 41

RPM Revoluciones por minuto. 4

RTC *Real Time Clock*. 6, 7, 11, 24, 27, 41, VII

SCL *Serial Clock*. 13–15, 27

SDA *Serial Data*. 13–15, 27

SPI *Serial Peripheral Interface*. 3

ST *Station*. 17, 41

TCP *Transmission Control Protocol*. 3, 16

UART *Universal Asynchronous Receiver / Transmitter*. 3, 15, 16, 41, VII

USB *Universal Serial Bus*. 28

V Voltio. 3, 4, 6–8, 10, 27, 28, 41, VII